ERDC/CRREL TR-17-6

US Army Corps
of Engineers®
Engineer Research and
Development Center

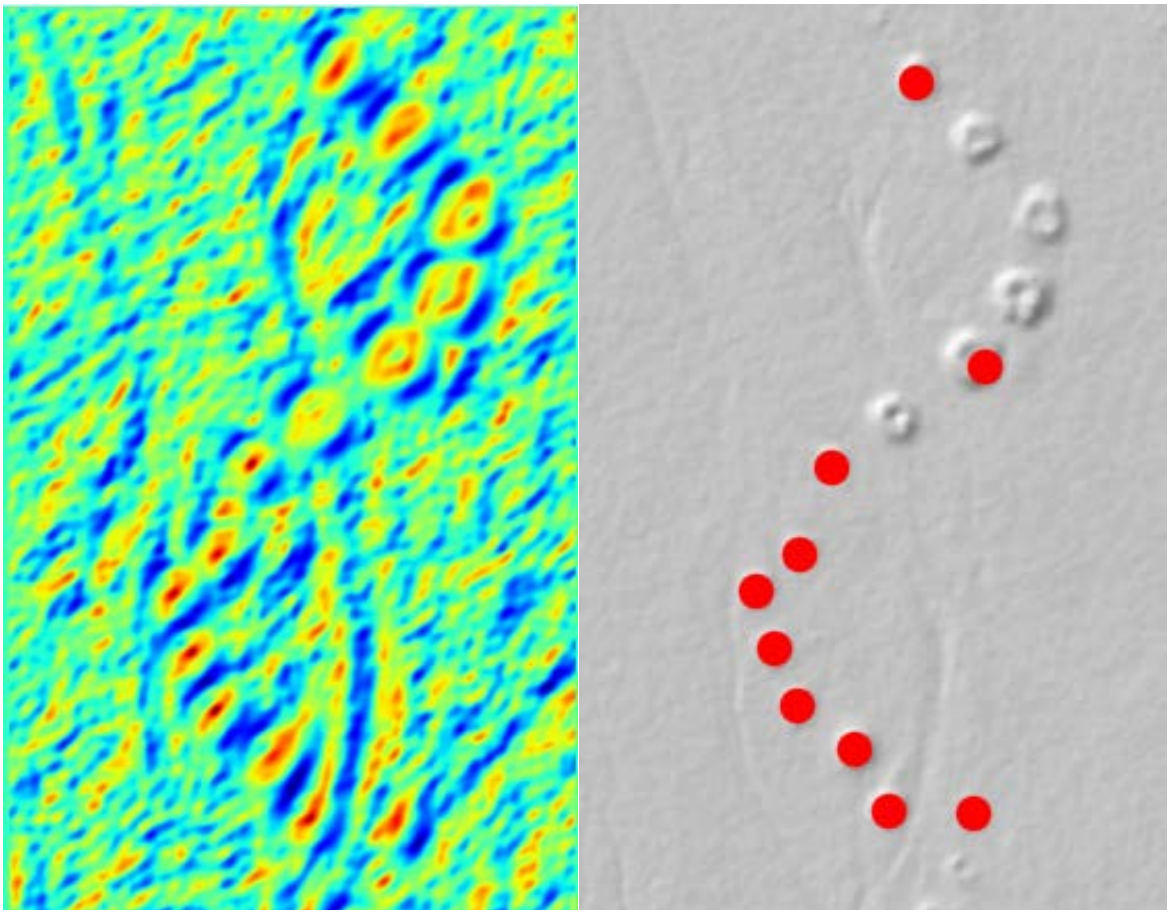
ERDC
INNOVATIVE SOLUTIONS
for a safer, better world

*Geospatial Remote Assessment for Ingress Locations (GRAIL)*

# Linear- and Repetitive-Feature Detection Within Remotely Sensed Imagery

Brendan A. West

April 2017



Cold Regions Research
and Engineering Laboratory

**The U.S. Army Engineer Research and Development Center (ERDC)** solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at http://acwc.sdp.sirsi.net/client/default.

ERDC/CRREL TR-17-6
April 2017

# Linear- and Repetitive-Feature Detection Within Remotely Sensed Imagery

Brendan West

*U.S. Army Engineer Research and Development Center (ERDC)*
*Cold Regions Research and Engineering Laboratory (CRREL)*
*72 Lyme Road*
*Hanover, NH 03755-1290*

Final Report



Prepared for Army Terrestrial Environmental Modeling and Intelligence System (ARTEMIS)
U.S. Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

Under Work Item 9K3D08 for the Geospatial Remote Assessment for Ingress Locations (GRAIL) Project

# Abstract

The United States Army has a variety of applications for identifying features of interest within remote imagery. Whether it is characterizing a landscape while planning operations or trying to find particular installations in an urban setting, the Army can glean a significant amount of information from imagery data. This study investigates methods that can detect linear and repetitive features contained in remotely sensed images that are in panchromatic or true-color formats. Image-processing techniques, including Hough transforms, machine learning, and template matching, are capable of detecting different kinds of features within images. However, the success of these methods depends on effectively preprocessing image data, which has proven difficult and intensive for certain images. In many cases, the amount of user interaction needed to produce useful results exceeds the amount of labor needed to manually inspect individual images. At their current state, these methods provide useful tools to help analysts detect features but do not replace their expertise. This report summarizes several techniques for preprocessing image data and then detecting linear and repetitive features in that data.

# Contents

# Figures and Tables

## Figures

## Tables

# Preface

This study was conducted for the Army Terrestrial Environmental Modeling and Intelligence System (ARTEMIS) program under Work Item 9K3D08 for the Geospatial Remote Assessment for Ingress Locations (GRAIL) project. The technical monitors were Dr. Sally Shoop (CEERD-RRH) and John Eylander (CEERD-RR).

The work was performed by the Terrestrial and Cryospheric Sciences Branch (CEERD-RRG) of the Research and Engineering Division (CEERD-RR), U.S. Army Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory (ERDC-CRREL). At the time of publication, J. D. Horne was Chief, CEERD-RRG, and Dr. Mark Moran was the Technical Director, CEERD-RZT. The Deputy Director of ERDC-CRREL was Dr. Lance Hansen, and the Director was Dr. Robert Davis.

The author wishes to thank Dr. Sally Shoop, Research Civil Engineer, Force Projection and Sustainment Branch, ERDC-CRREL, for her assistance with defining this project's research goals and for project management. The author also thanks Elke Ochs and Brian Tracy, Terrain and Ice Engineering Group, ERDC-CRREL, for processing and providing test imagery data, and Dr. M. Andrew Niccolai, Signature Physics Branch Chief, and Michael T. Ekegren, Signature Physics Branch, for their technical reviews of this work.

COL Bryan S. Green was the Commander of ERDC, and Dr. David W. Pittman was the Director.

# Acronyms and Abbreviations

| | |
|---|---|
| ARTEMIS | Army Terrestrial Environmental Modeling and Intelligence System |
| BRIEF | Binary Robust Independent Elementary Features |
| CMYK | Cyan, Magenta, Yellow, Key |
| CRREL | Cold Regions Research and Engineering Laboratory |
| ERDC | U.S. Army Engineer Research and Development Center |
| FAST | Features from Accelerated Segment Test |
| FHL | Fort Hunter-Liggett |
| GIS | Geographic Information System |
| GRAIL | Geospatial Remote Assessment for Ingress Locations |
| HSV | Hue, Saturation, Value |
| Lab | Lightness, a (Green–Red), b (Blue–Yellow) |
| LoG | Laplacian of Gaussian |
| LZ | Landing Zones |
| ORB | Oriented FAST and Rotated BRIEF |
| RGB | Red, Green, Blue |
| SIFT | Scale-Invariant Feature Transform |
| SURF | Speeded-Up Robust Features |
| SVM | Support Vector Machines |

# 1 Introduction

## 1.1 Background

The Army desires the ability to deliver cargo, equipment, and personnel to harsh locations almost anywhere on the planet. This requires locating areas that are large, flat, and obstruction-free with sufficient soil strength to support at least one aircraft landing and taking off (Ryerson and McDowell 2008). The goal of Geospatial Remote Assessment for Ingress Locations (GRAIL) is to develop tools that will help the Army remotely identify these types of locations where an aircraft can land without any prior ground activity or surveillance from Army personnel. The GIS- (geographic information system) based GRAIL Toolkit provides a capability to identify potential landing zones (LZs) that have acceptable slope and land cover. However, field testing has shown that some of these LZs contain telephone wires and poles, fences, well holes, etc., that do not show up in the GIS analysis. This paper introduces image-processing techniques and tools that may help detect some of these features in remotely sensed imagery.

## 1.2 Objectives

This report explores methods for identifying two types of features that pose issues for aircraft attempting to land or takeoff at a particular site or for ground forces passing through an area. Linear and repetitive features correspond to infrastructure that spans across large spatial lengths, as opposed to isolated features such as boulders or houses. Linear features consist of roads, fence lines, cracks in ice, and other contiguous entities. Repetitive features are those that have recognizable characteristics and follow general patterns but are not visibly connected, at least from a remote perspective. An example of these is telephone poles. These features may not appear linear in the remote imagery, but still pose issues for an aircraft attempting to land in an area where they are present.

## 1.3 Approach

This report serves as a summary of image-processing methods and techniques that could help identify these features of interest. It contains initial methodologies, discussions, and examples of how to detect both linear and repetitive features within remote imagery. There is certainly more to explore on this topic, and some of these areas are introduced in Section 6:

"Future Directions." The ultimate goal of this study is to identify methodologies that are as autonomous as possible so as to reduce user input. Current techniques used by the Army and other entities are time-consuming and require individuals to survey imagery and manually identify features.

This study used Matlab, Python, and several of their add-on packages for image processing. Matlab's Image Processing, Computer Vision System, and Statistics and Machine Learning Toolboxes provide several useful tools and modules for these analyses. Similarly, the following Python packages supplemented the base Python package (version 2.7): Numpy, OpenCV, Scikit-Image (Skimage), and Scipy. This report contains references to language-specific tools or functions. This does not mean those capabilities cannot be used in other programming languages but that additional programming may be necessary to transfer them into those languages.

## 1.4 Test cases

This write-up uses a few test cases to illustrate the described concepts. The text will reference these test cases and will highlight them in their own sections. These are examples only and do not represent the optimal methods and steps used to detect linear or repetitive features, and the imagery used in these examples is limited to panchromatic and RGB images. There is one linear-feature detection test case, which attempts to identify fence lines within an image of a landscape. There are two repetitive-feature test cases that use different methods to identify telephone poles and Karez holes. A Karez is an underground aqueduct system used in many Middle Eastern countries to connect remote villages to a water resource. These systems are characterized by well holes that connect the underground tunnel to the ground surface. The fence line and Karez hole figures are hillshade images processed from BuckEye elevation data collected in a region of Afghanistan. It is important to note that the hillshade images are used only to illustrate the image-processing techniques described in this report. Direct use of elevation data provides much more information about a particular site than a two-dimensional image does. The telephone pole case includes Google Map images for two field sites (JM-12 and JM-13) at Fort Hunter-Liggett (FHL), an Army base in California. These two field sites were determined as sufficient LZ's through GIS analysis; however, a site visit revealed there were telephone wires running through these locations, which the GIS tools failed to identify. This is an example of how the feature-detection methods may supplement the rest of GRAIL functionality.

# 2 Data Preprocessing

Many feature-extraction methods require some level of preprocessing to enhance the image data or highlight features while reducing sections of the data that are not of interest. This may include altering the image's contrast or intensity range, sharpening or blurring the data, segmenting the image, or many other alterations. The preprocessing method used for a particular image and its effectiveness is usually specific to the type of image being refined, the image characteristics, the atmospheric conditions at image capture, the perspective of the image, and the resolution-to-object ratio of the image. What works well for one image often does not work for another. The ultimate goal of preprocessing is to prepare the image data in such a way that feature-extraction algorithms can then function more effectively on that image. Not all detection methods require preprocessing, such as the cross-correlation method for finding repetitive features; however, applying some form of preprocessing often enhances detection-method performance across all detection methodologies.

Image enhancement methods fall into two categories—those that work in the spatial domain and those that function in the frequency domain. Spatial-domain methods are alterations applied locally to the image pixels whereas frequency-domain methods are performed globally and require transforming the image into the frequency domain through a Fourier transform (Maini and Aggarwal 2010; Rahnama and Gloaguen 2014). An inverse Fourier transform converts the image back after the spatial methods are completed.

Many extraction methods, such as machine learning, require segmenting the image into distinct regions to work properly. However, this can be a challenging task. Gonzalez and Woods (2002) state that proper segmentation is one of the most difficult tasks in all of the image-processing field and that feature detection is more likely to succeed with accurate segmentation. There has been substantial work in this field to develop segmentation algorithms and methods for a variety of applications. Although this report does not contain an exhaustive list of methods, it explores some of the main themes in that area.

## 2.1 Spatial-domain operations

The majority of spatial-domain operations require images in an intensity format where each pixel is represented with a grayscale value ranging between 0 (black) and 1 (white) or a binary format where pixels are either 0 or 1. There are several spatial-domain techniques; however, two common categories of these techniques are point processing, which alters pixel characteristics individually, and histogram manipulation, which alters an image based on the distribution of intensity values within the whole image (or a reference image). An example of a point-processing method is taking the negative of an image. An example of histogram manipulation is rescaling a light-colored image to the lighter range of the grayscale so that its details are more pronounced.

Morphological operations are common point-processing methods that can reduce noise in binary image data (Efford 2000). These operations produce a new image after comparing an initial image with a structure element, or "kernel," which is a small matrix of pixels with 0 or 1 values that define a shape, such as a diamond, cross, circle, etc. (Efford 2000). Table 1 represents an example of a 5 × 5 diamond structure element. The central pixel of the structure element is passed over each pixel in the input image, and the pixels with a value of 1 in the structure element are compared with the corresponding pixels in the image. Different operations test how well the image pixels and structure element match:

- The element "fits" the image if each 1 pixel in the structure element corresponds to a 1 pixel in the image.
- The element "hits" the image if at least one of the 1 pixels in the structure element corresponds to a 1 pixel in the image (Quackenbush 2004).

Table 1. Example of a 5 × 5 diamond structure element. The central *red* pixel indicates where the element lines up with a specific pixel in the image.

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |

Morphological operations are used for two fundamental processes—erosion and dilation. As the name suggests, the erosion process erodes pixels

from the edge of features, which is used to reduce noise and removes small pixel groups. If there is not a complete "fit" between the structure element and a region in the input image, then the central pixel of that region is made a 0. In dilation, if there is a "hit" between the structure element and the input image region, then the corresponding pixel is made a 1. This process adds a layer to the edges of image features and fills any small gaps. The effects of these operations depend on the size and shape of the structure elements used. These fundamental operations can combine to form useful compound morphology operations (Quackenbush 2004). For example, the "area-opening" operation removes small outlier groups of edge pixels that are not a part of actual features and is achieved by an erosion followed by a dilation with the same structure element. This process opens up gaps between features connected by small pixel bridges. The closing operation does the opposite of the area-opening operation and connects any small gaps between features. There are a variety of other operations that produce different preprocessing results (Quackenbush 2004).

Additional spatial filters, including the Ford filter (Table 2), directional sun-angle filters (Table 3), gradient filters, and Laplacian filters, use structural elements with various cell values to alter the image's pixels or to enhance linear features (Mavrantza and Argialas 2003).

Table 2. A 5 × 5 Ford filter intended to enhance details relevant to manmade objects and to reduce details of natural background. This is considered a high-pass spatial filter (Mavrantza and Argialas 2003).

| | | | | |
|---|---|---|---|---|
| −0.3 | −0.3 | −0.3 | −0.3 | −0.3 |
| −0.3 | −0.3 | −0.3 | −0.3 | −0.3 |
| −0.3 | −0.3 | 9.7 | −0.3 | −0.3 |
| −0.3 | −0.3 | −0.3 | −0.3 | −0.3 |
| −0.3 | −0.3 | −0.3 | −0.3 | −0.3 |

Table 3. A 3 × 3 directional sun-angle filter that takes into account the direction of the sun and is intended to enhance edges affected by shadowing (Mavrantza and Argialas 2003). This example filter is for sun from the Northwest.

| | | |
|---|---|---|
| −1 | −1 | 0 |
| −1 | 0 | 1 |
| 0 | 1 | 1 |

Median filtering provides another method for enhancing the image data, specifically for reducing noise. Similar to structure elements, a window surrounding a specified number of pixels moves through the image. The median intensity value within that window is then applied to the target pixel, which smooths intensity values across the image (Fisher et al. 2003). The same idea can be applied with other statistical values, such as mean, maximum, or minimum of the image region. Median filters are particularly useful for situations where impulse noise is present, which is noise that appears as small black and white dots overlaid on the image (Gonzalez and Woods 2002).

In conclusion, there are multitudes of spatial-domain operations available to use. Most image-processing software programs have a number of these operations preprogrammed or assembled in packages for use. However, it is important to know that a certain amount of trial and error is necessary to settle on the appropriate approach to spatial-domain image processing.

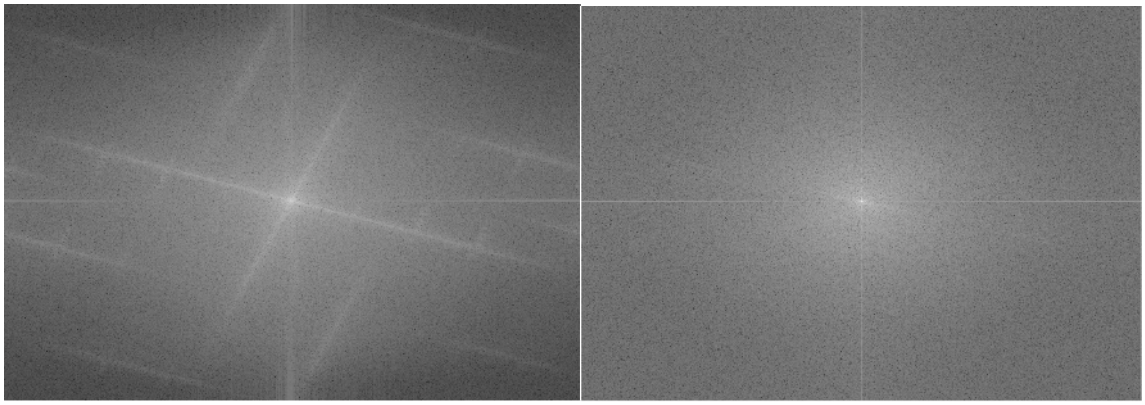
## 2.2 Fourier-domain operations

Frequency-domain filters work on image data that have gone through a Fourier transform, in which pixel-by-pixel data are converted into a series of sinusoidal functions of varying frequencies and amplitudes that approximate the pixel field. The boundaries of distinct features tend to have higher frequencies than regions of the image that are continuous in intensity. A sudden change in intensity values between pixels is akin to a periodic function with high frequency, where the signal output changes from a low to high value quickly with respect to location in the image. Filter functions are based on the assumption that noise in the data is associated with high frequencies and are designed to target or ignore these frequencies depending on the intention of the filter (Fisher et al. 2003).

Images containing natural objects tend to have different Fourier transform characteristics than those with manmade objects, which may prove useful when trying to distinguish manmade structures, such as telephone poles and fences, from a natural background. Manmade structures typically have strong edges oriented in vertical or horizontal directions whereas natural structures do not. The strong features of manmade structures produce distinct lines in the image representation of the Fourier transform (Figure 1a) while natural objects produce less coherent images due to their lack of distinct lines and features (Figure 1b) (Wittman 2010).

Figure 1. A Fourier transform analyses of overhead images of CRREL buildings and of a forested area with few house and roads.

(a) CRREL Buildings (b) Forested Area



Frequency filters sometimes have equivalent functions in the spatial domain that are often less computationally expensive to use (Fisher et al. 2003). For example, gray-level slicing is the spatial domain equivalent to passing a band-pass filter over the data (Maini and Aggarwal 2010). Similarly, the Ford filter is the equivalent of a high-pass frequency filter. However, it is important to note that the equivalent spatial filters are only approximations of the frequency filters (Fisher et al. 2003).

As a reminder, high-frequency data is attributed to image noise or image regions where there are sudden changes in intensity (e.g., feature edges), whereas low-frequency data is synonymous with pixels that are consistent and do not vary much (e.g., image background). A low-pass filter removes high-frequency noise but permits low-frequency data, which smoothens the image. This is useful when an image has a lot of noise. A high-pass filter removes low-frequency information but permits high-frequency data, which sharpens features such as edges or lines. This is useful when the image contains a lot of unwanted background data. A band-pass filter retains data within a range of specified frequencies. This type of filter may be useful when there is noise and background data to remove from an image.

Low-pass filters sometimes create a visual effect due to the Gibbs phenomenon, where a series of artificial rings form around edges, which may prove problematic for subsequent linear-feature extraction (Wittman 2010). Gaussian filters work by passing a Gaussian signal over the data that smoothens any spikes or noise and avoids the Gibbs ring effect. The Butterworth filter represents a “discrete approximation to the Gaussian”

and has a lower computational cost than the Gaussian filter when implementing wide high- or low-pass filters. The Gaussian filter is better suited for high- or low-pass filters with narrow pass ranges (Fisher et al. 2003). The intent of these filters is to reduce the impact of noise in the data by effectively blurring the image, including feature edges.

A high-pass filter or an "unsharp filter" sharpens image data by subtracting a smoothed version of the image from itself. The result is called an "edge" image and emphasizes the high-frequency regions in the original data. This is then added back to the original image to produce a sharpened version of the image (Fisher et al. 2003).

The intention of these different filters is to improve the image data before they are passed through feature extraction methods. There are many methods to achieve different image enhancements, and the most useful methods may depend on different aspects of the input image, such as terrain type, ground cover, etc.

## 2.3 Segmentation

As stated previously, proper image segmentation is paramount to effective feature-extraction methods. This process is not straightforward, and a variety of approaches achieve useful results. The result of many of these methods is binary images where the segmented regions are pixels with a value of "1" and the background pixels have a value of "0."

### 2.3.1 Intensity-based thresholds

One of the simplest ways to segment an image is based on the distribution of intensity values in that image. When features of interest appear lighter or darker than the rest of the image, an intensity-threshold value may adequately separate the features from the background. This value can help to convert the input image to a binary image where anything above the threshold is converted to white and anything below is converted to black. If the image is initially in a color format, then the image must be converted to an intensity format before thresholding in this manner.

The threshold value can be determined manually by inspecting the intensity distribution of the image, or it can be determined automatically. Two common ways to do this are with Otsu's method and K-means clustering.

#### 2.3.1.1 Otsu's method

Otsu's method identifies a threshold value that minimizes the spread between the intensity distributions of light and dark features in an image (Morse 2000). There are two variations to Otsu's method: *global*, which considers the intensity distribution of the entire image, and *local* or *adaptive*, which considers only the intensity values of a certain neighborhood of pixels as the algorithm works through the image. The global variation proved more useful in the test cases, which is likely because the features of interest and background data were fairly consistent within each test case image. For example, the intensity values of the telephone pole shadows were relatively consistent from one shadow to the next. However, small variations in the background data made the local intensity distribution more variable from one neighborhood to the next. Including a larger intensity dataset likely reduced variability in the Otsu threshold calculation, thereby rendering the results of the global application more reliable than a localized approach. However, the effectiveness of the Otsu threshold depends on the image's intensity distribution.

#### 2.3.1.2 K-means clustering

K-means clustering tries to separate the intensity data into different groups by selecting an intensity value "such that each pixel on each side of the threshold is closer in intensity to the mean of all pixels on that side of the threshold than the mean of all pixels on the other side of the threshold" (Morse 2000). This is an iterative process where the updated threshold is the value halfway between the intensity averages of the points on either side of the old threshold. This process continues until the threshold value converges. K-means clustering works well for situations where the respective intensity distributions of light and dark regions are approximately equal but struggles when the distributions vary significantly (Morse 2000). An advantage to this method is that the user can easily segment the image into more than two groups.

#### 2.3.1.3 Intensity-threshold issues

Otsu's method and K-means clustering both struggle in a few situations. For example, both methods are less effective when the feature of interest falls somewhere in the middle of the intensity distribution. However, one way around this is to redistribute the intensity values of the image towards

one end of the spectrum before implementing either method. The advantage to K-means clustering is that the user may specify enough groups to segment that one of them captures the feature of interest before any intensity redistribution is required.

Another area where these methods break down is when the feature itself has a range of intensity values. An example of this is when segmenting Karez holes in hillshade images. Hillshade images present elevation data in a way that humans can easily understand. This data is represented as if there was a light source located to the northwest with an altitude angle of 45°*, which produces an image with shadows and light regions around each elevated feature in the data. However, this means the mounds corresponding to Karez holes have both a light and dark side, which makes it difficult to segment the entire Karez feature based on its intensity values. A way around this is to pass a gradient filter over the image, which calculates the gradient of the intensity field. This is useful because the changes in intensity from the background terrain to the light and dark sides of the Karez holes are nearly the same. The result of the gradient filter is a new intensity image where the outlines of the Karez holes have similar intensity values with which Otsu's method or K-means clustering may work. This example illustrates a common trend with these segmentation techniques: get the image data in an intensity format where the features of interest are similar values that are distinguishable from the background, and then implement Otsu's method or K-means clustering to convert the data to a binary dataset.

### 2.3.2 Watershed segmentation

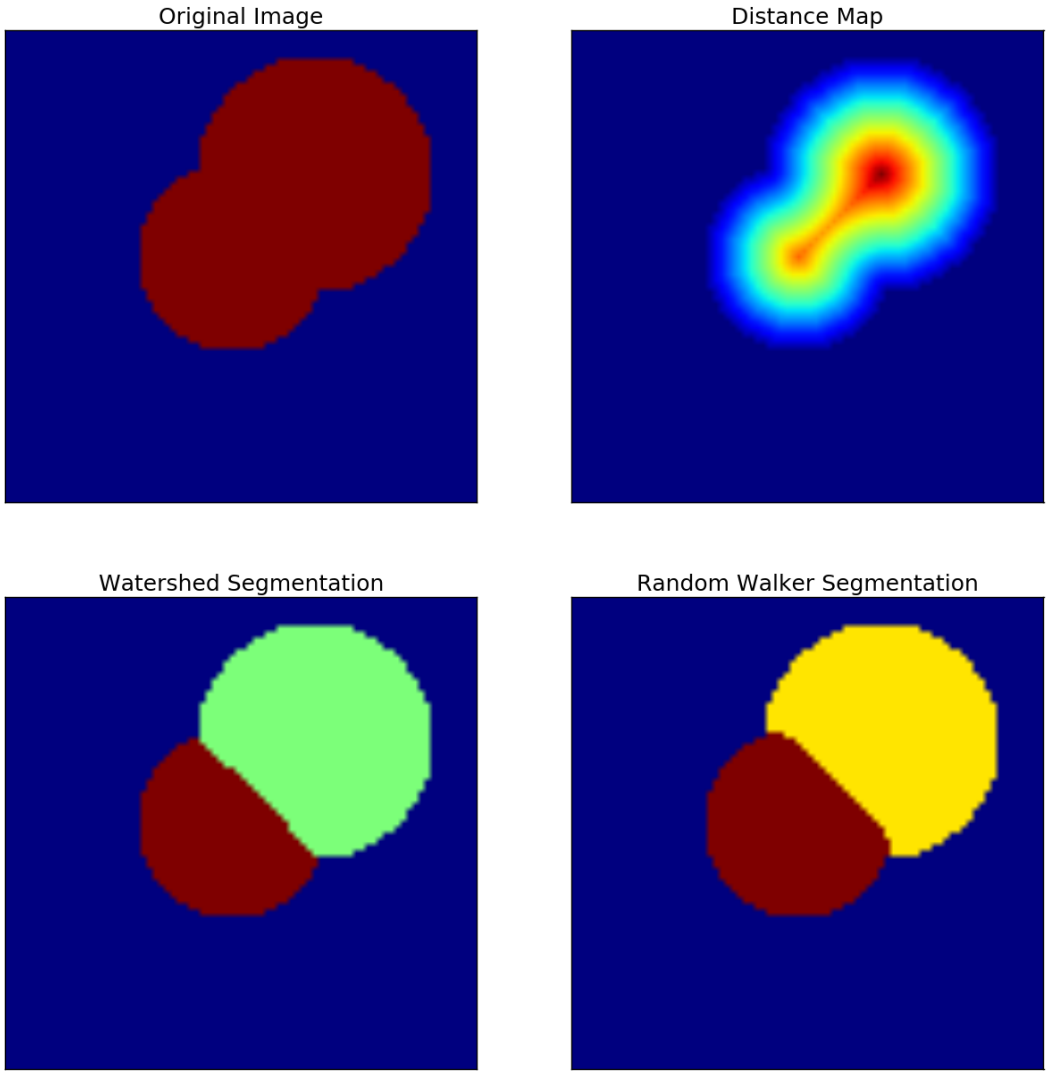
This method is named for a hydrological analogy that explains the principles of the algorithm. In this analogy, a binary image is treated as a terrain map where dark areas are low-elevation catchment basins and white areas are high-elevation areas. In the real-world scenario, water flows from high ground and collects in catchment basins. Watershed lines dictate the regions of terrain that drain into the same particular basins, effectively segmenting one basin from others (Eddins 2002).

The watershed lines in an image are found with the following procedure. First, the distance is calculated from each "basin" pixel to the closest "edge" pixel, where black and white pixels interface. An example of this

* To convert to radians, multiply the angle by 0.01745329.

distance map is the top right image in Figure 2. The local maxima of the resultant distance map are used as seed points from which each "basin" is "flooded." The watershed lines are where different flood sources meet. Occasionally, this process results in oversegmentation, at which point segmented regions must be combined with a different process (Beucher 2010).

Figure 2. Examples of different segmentation schemes and results.




This method has issues when identifying features that are small in relation to the whole image size or features that vary in size from one instance to the next. In addition, it is based on the local intensity values of the image, which means it suffers from the same problem that Otsu's method does. It

fails to identify regions of interest that are not overly bright or dark. However, redistributing the intensity values of the image before implementing the watershed algorithm may help.

### 2.3.3 Random walker segmentation

Random walker segmentation is based on the probability that a random path around an image will cross a specific target segment before it reaches any other segment. The user defines a set of seed points that are associated with each type of region ("feature" and "background" in this case). There are different ways to choose these seed points, including the same way as the watershed method (local maxima of a distance map). The algorithm then calculates the probability of a random walker leaving each unseeded pixel and finding one of the seeded points (Grady 2006). The result is a probability map depicting how likely each unseeded pixel is associated with each seed pixel. During testing, there was not a large difference between the results of the watershed and random walker segmentation techniques. Figure 2 illustrates and compares the results of watershed and random walker segmentation for basic shapes. The segmentation results for both methods show similar results in this example as illustrated in the bottom images of Figure 2.

# 3 Linear-Feature Extraction

There are a wide variety of techniques and algorithms used to extract linear features from images, including mathematical morphology, Hough transforms, multiresolution techniques, and many more. The following literature review is not exhaustive given that a number of extraction techniques are proprietary (Quackenbush 2004) or are not readily implemented in the software chosen for this report—Matlab or Skimage.

A number of feature-extraction techniques process and compare the image data against models that describe the important characteristics of a feature or object. These feature models are either "rigid" or "flexible." A rigid model sets strict values or guidelines for the feature's shape, maximum size, or spectral characteristics. Flexible models set values for less specific properties, such as curvature, symmetry, homogeneity, etc. (Quackenbush 2004). The function or process used to compare the image data with the model is what varies between different techniques.

Feature-extraction methods are usually split into three steps: edge detection, edge tracking, and linking edges. Edge detection consists of finding possible feature edges in the image via the algorithms explained next. Edge tracking entails following the possible edges once they are identified. Edge linking connects edges that most likely define a feature and estimates missing edge pixels (Rahnama and Gloaguen 2014).

## 3.1 Edge detection

### 3.1.1 Morphology operations

The morphology operations discussed in the preprocessing section can also help to detect features and edges within an image. Regions of the image where these linear-feature structure elements "fit" represent linear features or the edges of features. Often using these different methods in conjunction will provide different results.

### 3.1.2 Gradient and Laplacian operators

Calculating the gradient of an image field indicates the direction of largest change in intensity. This is useful for detecting edges as the interface between features and the background are usually characterized by large changes in intensity. A Laplacian operator takes this process another step

and looks at how quickly the intensity values change within the field. Because of the Laplacian operator's sensitivity to noise, passing a Gaussian filter over the data can help to reduce noise before implementing the Laplacian. It is possible to combine the Gaussian and Laplacian steps into a single process called the Laplacian of Gaussian (LoG) filter, which reduces the number of operations performed on the data (Fisher et al. 2003). The zero-crossing points of the resultant Laplacian indicate the locations of an edge; however, some of these zero-crossing points correspond to minor edges or image texture. Taking the variance of intensity values near each detected edge estimates how significant that particular edge is. A small variance indicates a minor edge, and a large variance indicates a significant edge. Specifying a variance threshold for this process gives us another parameter to fine-tune the linear-feature extraction process. Any edges below the variance threshold are ignored to remove minor features during the extraction process (Claypoole et al. 1997).

The spatial field representation of an image is a discontinuous function, which cannot be differentiated. This poses an issue when trying to calculate the gradient or Laplacian of that image, which represent the first- and second-order spatial derivative of the image, respectively (Fisher et al. 2003). To overcome this problem, these operations are fulfilled with discrete filters that approximate the first-order or second-order derivatives. Table 4 and Table 5 illustrate some of these structure elements. It is important to understand that these gradient and Laplacian methods are only approximations of the true derivatives of the image; however, they produce good results.

The Roberts cross operator performs best when edges are oriented 45° to the pixel array, and it is very quick to compute since it uses only a 2 × 2 kernel. However, this method does not deal well with noise (Fisher et al. 2003).

In the case of vertical and horizontal filters, they are both applied simultaneously throughout the image (Vilnius University 2009). Table 6 is an example of how these filters are applied to an image (Delmas 2015). Take an input image, *X*, and pass the vertical Prewitt filter (Table 4) over it to get an altered output image, *Y*. The red box in *X* reflects where the filter structure element is located in the input image. The red cell in *Y* reflects the target pixel in the output image.

Table 4. Gradient filters approximate the gradient of the image intensity field. There are multiple gradient filter types including Prewitt, Sobel, and Roberts cross (Mavrantza and Argialas 2003; Vilnius University 2009).

Vertical and horizontal Prewitt filters

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Vertical and horizontal Sobel filters

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

| | | |
|---|---|---|
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Roberts cross operators

| | |
|---|---|
| 1 | 0 |
| 0 | −1 |

| | |
|---|---|
| 0 | 1 |
| −1 | 0 |

Table 5. A Laplacian filter approximates the Laplacian of the image intensity field. As with the gradient filter, there are multiple types of Laplacian filters (Fisher et al. 2003)

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

Table 6. Example of applying a Prewitt filter to simulated image data.

| | | | |
|---|---|---|---|
| $X_{11}$ | $X_{12}$ | $X_{13}$ | … |
| $X_{21}$ | $X_{22}$ | $X_{23}$ | … |
| $X_{31}$ | $X_{32}$ | $X_{33}$ | … |
| … | … | … | … |

| | | | |
|---|---|---|---|
| $Y_{11}$ | $Y_{12}$ | $Y_{13}$ | … |
| $Y_{21}$ | $Y_{22}$ | $Y_{23}$ | … |
| $Y_{31}$ | $Y_{32}$ | $Y_{33}$ | … |
| … | … | … | … |

Apply a Prewitt filter to $Y_{22}$:

$$Y_{22} = -X_{11} + X_{13} - X_{21} + X_{23} - X_{31} + X_{33} \quad (1)$$

Continue this process through the rest of the output-image pixels, and then repeat for the horizontal Prewitt filter. Simply add together the magnitudes of the vertical and horizontal filter results for each pixel. Sometimes the results of this calculation are outside the 0–1 range typical for

intensity values, at which point the data is scaled to set the new minimum equal to 0 and new maximum to 1.

### 3.1.3 Additional methods

There are additional methods for detecting edges, but those mentioned here are techniques readily implemented in Matlab or Skimage. Examples of other methods include multidimensional gradients; squared local contrast; and the compass operator, which uses several orientations of a Sobel (or other) filter to detect edges running in multiple directions. Additional information on these operators and others can be found in Burger and Burge (2009, 2013).

All of these edge-detection methods do exactly what their name implies—detect potential edges—but they do not necessarily connect edge segments that likely represent the same line. This step usually requires a feature-extraction technique.

## 3.2 Feature extraction

### 3.2.1 Canny method

The Canny method combines some of the procedures already mentioned and includes additional edge-tracking and linking steps. The process starts with Gaussian smoothing and then passes a gradient filter over the image (Fisher et al. 2003). The method takes in two parameters: a threshold for the minimum edge strength in terms of gradient magnitude and a threshold that is used in tracing edge sections. The gradient must be higher than the first threshold to indicate a potential edge section. If the gradient values in the pixels adjacent to these edge sections are greater than the second threshold, the algorithm will assume those pixels are supposed to be edge sections, also, and will merge them with the initial strong-edge pieces (Wang et al. 2006). The output of this method is a binary image outlining the edges of features in the initial image (Fisher et al. 2003). This method is often used as an edge-detection step for other feature detection methods, such as the Hough transform, which is described later.

### 3.2.2 Rothwell method

The Canny method sometimes fails to handle corners where different edge segments come together. In some cases, the gradient direction can be as much as 70° off from its true direction. The unreliability of this direction

poses an issue when Canny classifies edges and non-edges; and in many cases, it completely ignores these corner edge segments. The Rothwell method is similar to Canny but includes the topology of the image in an attempt to capture the corners of features (Rothwell et al. 1995).

The Rothwell method conducts Gaussian smoothing and then identifies potential edges in a similar fashion to Canny. In some cases, the edges are located with a zero-crossing method instead of a gradient-based method. The edges identified in this step are considered only a base set, and vertices are added to this set as they are found. The algorithm places vertices at edges that "have either only a single neighbour (in which case they represent the end of a dangling [edge] chain), or are [edges] which have more than two [edges] connected to them (junctions)" (Rothwell et al. 1995). The "single neighbour" refers to a pixel adjacent to only one other pixel denoted as an edge, which indicates the end of that line segment. After these vertices are determined, the algorithm follows the connected edges and records them to a list so as to characterize all the edges and connecting vertices as a single entity (Rothwell et al. 1995).

### 3.2.3 Hough transform

The Hough transform is a versatile method that can identify features of different shapes, or those with gaps in their boundary, and is relatively tolerant of noisy image data. It requires an edge-detection method prior to running (Fisher et al. 2003). The following explanation follows an example from Hamarneh et al. (1999). Consider intersecting lines in the Cartesian space in Figure 3.

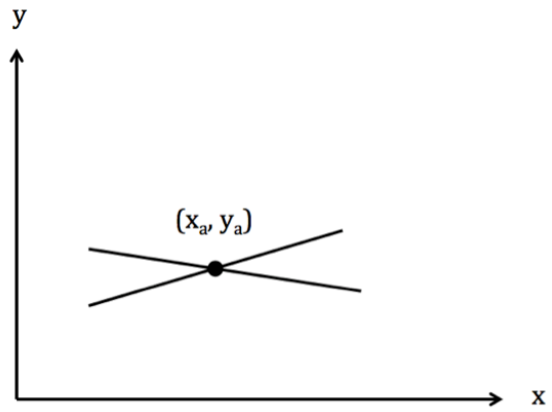
Every line that passes through point $(x_a, y_a)$ satisfies the following equation:

$$y_a = m \cdot x_a + c \tag{2}$$

where

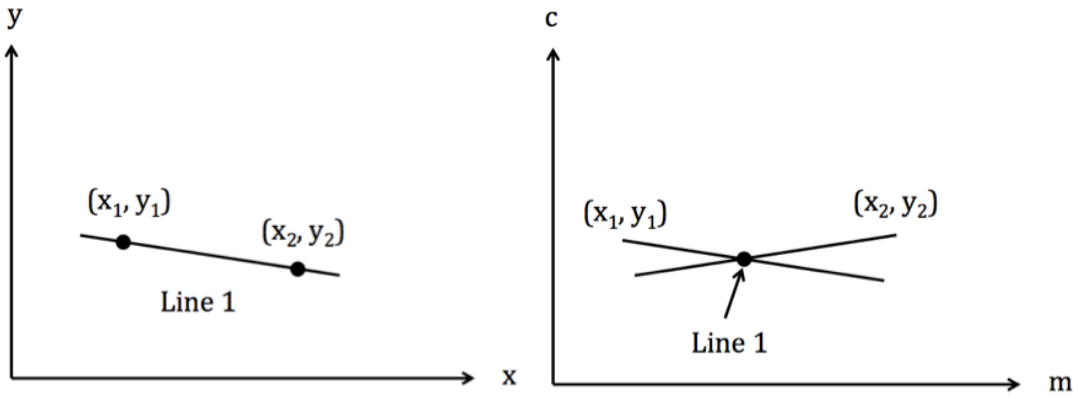
$m$ = slope of a particular line and
$c$ = y-intercept of that same line.

Figure 3. Cartesian coordinates and line intersection.




Hamarneh et al. (1999) show that a c-m coordinate system can be defined in which all the lines that cross through a Cartesian point $(x_a, y_a)$ can be represented as a single line in c-m space. Consider two pixels in Cartesian space in Figure 4.

Figure 4. Transformation of a Cartesian line to a c-m framework.




For each point in the left image of Figure 4, one can define a line in the c-m frame that represents all the Cartesian lines that intersect that particular point. Now consider a line in the Cartesian framework that passes through both points; this corresponds to an intersection of two lines in the c-m space. This implies that all points (image pixels) that lie on a single Cartesian line correspond to lines in the c-m system that cross at a single point. In other words, collinear segments in the Cartesian system will correspond to a single point in the c-m system. This is useful for identifying lines that do not appear contiguous because of gaps though they are contiguous in reality (Hamarneh et al. 1999).

The rest of this process includes converting the c-m system into a two-dimensional matrix and comparing it with the results of the prior edge-detection process. This provides a histogram that indicates the frequency of detected edges corresponding to points lying on the same line. High-frequency values in this histogram relate to lines in the image data. Detection of different shapes is possible as long as a parametric equation can describe the shape of interest. For example, a circle requires using a three-dimensional matrix of the circle's radius and center coordinates (Hamarneh et al. 1999).

A disadvantage of the Hough transform is that the detected lines in the c-m system are infinite. If there are unrelated features that happen to line up in the image, the Hough transform may detect them as a single line. In addition, a problem arises when using the representation of a line as in Equation (2) because vertical lines correspond to an infinite slope. To avoid this, use the parametric definition of a line as described in Equation (3):

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta) \tag{3}$$

where

$\rho$ = the distance from a line to the coordinate system origin and
$\theta$ = the angle between the $\rho$ vector and x-axis of the coordinate system.

With this representation, the Cartesian lines are represented as curves in the parametric space (Hamarneh et al. 1999).

Fitton and Cox (1998) discuss a multiscale method of applying the Hough transform. This method consists of identifying large linear features in a full-size image, removing those features from the data, then halving the image and repeating the process. This is continued until the image segments include the end points of the smallest linear features that are of interest. The idea of reducing the scale of the image is from Mirmehdi et al. (1991). The process of removing features from the data simplifies subsequent passes with the Hough transform (Fitton and Cox 1998).

Skimage provides a probabilistic Hough transform that reduces the number of intersection points considered during calculations, which speeds up

the process when compared with the normal Hough transform. A random subset of intersection points is selected, and then line features are identified by moving along these selected components in the Cartesian space (Scikit-Image 2011). This Skimage function provides parameters to specify the minimum line length and maximum gap between linear features, which allows the program to combine linear segments that are close to each other. The function also lets the user specify a minimum strength threshold that each intersection point must be to be considered for the remaining calculations.

## 3.3 Comparison of techniques

There are a few references that compare edge detection algorithms against each other. Musoromy et al. (2010) summarize results of other comparative studies and show that for the methods presented in this review, Canny performs better than Rothwell, both perform better than LoG, and LoG performs better than Prewitt and Sobel for noisy image data. In terms of compute time, Canny was the fastest of the methods, followed by Sobel, Laplace, and Rothwell (Musoromy et al. 2010). However, these time results depend on how many images are processed and the amount of preprocessing. The time differences this paper presented are on the order of ≈10 ms for processing 45,000 images.

Argialas and Mavrantza (2004) compare edge detection methods against a Hough transform method. Similar to the first reference, this group found that Canny provides the best results, followed by Rothwell. Argialas and Mavrantza (2004) find that the Hough transform does not locate and extract edges as accurately as the edge-detection methods do and that its results are highly dependent on the input parameters.

Wang et al. (2006) find similar results, stating that Sobel, Canny, and Rothwell have similar performance to each other, with Sobel producing slightly worse results than the other two and LoG performing worst. However, they say the difference in performance between these methods is negligible by their standards. This study also states that all of these methods are highly dependent on their input parameters, and their performance with fixed parameters is considerably lower than with dynamic parameter inputs. Wang et al. (2006) also provide results illustrating how method performance depends on the analyzed image. Some methods perform best for some images while other methods perform better for other images. It seems better to use methods and parameters optimized for each

image instead of general methods and values. As a side note, they also suggest increasing the default Canny threshold in Matlab by a factor of two (Wang et al. 2006).

It appears from these comparative studies that the Canny and Rothwell methods perform better than the others. It makes sense that they perform similarly because Rothwell is based on the Canny method. Rothwell et al. (1995) claim that the Rothwell method recognizes and characterizes edge-corners better than the Canny method. However, the other studies seem to suggest that Canny is better at extracting the actual linear features and takes less time than Rothwell. It is uncertain whether the corner-extraction ability that Rothwell provides is necessary for Army feature detection applications or if Canny is sufficient.

## 3.4 Linear-feature detection methodology

This section highlights an example methodology for detecting linear features in imagery. For this test case, the base image is an RGB image of farmland, which is converted to grayscale and then processed with a series of gradient and Gaussian filters to improve the data. A Canny edge detector then highlights edges, which are fed into a Hough transform that completes the feature detection.

### 3.4.1 Linear-feature methodology

1. Acquire the base image (Figure 5).

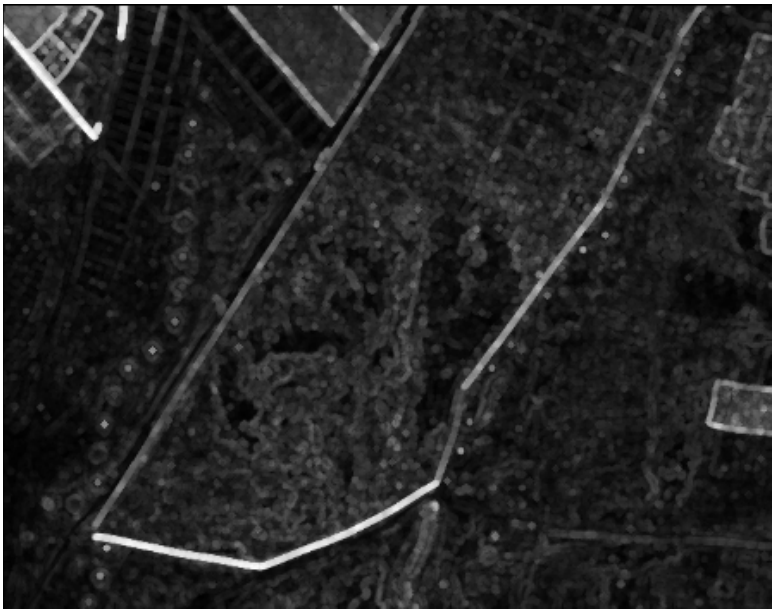
Figure 5. Farmland with fences.

2. Convert the base image to grayscale (Figure 6).

Figure 6. Conversion of base RGB image to grayscale.

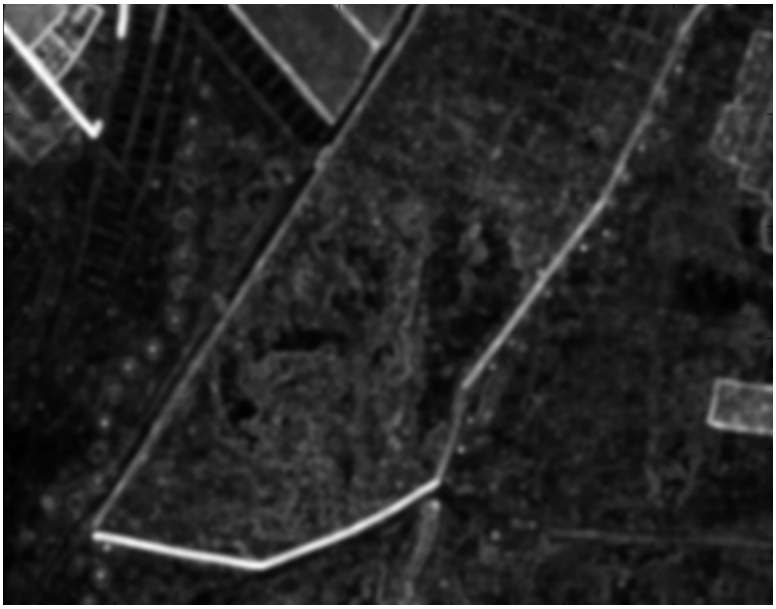
3. Pass a gradient filter over the grayscale image to highlight changes in pixel intensity (Figure 7).

Figure 7. Result of passing gradient filter over the grayscale image.



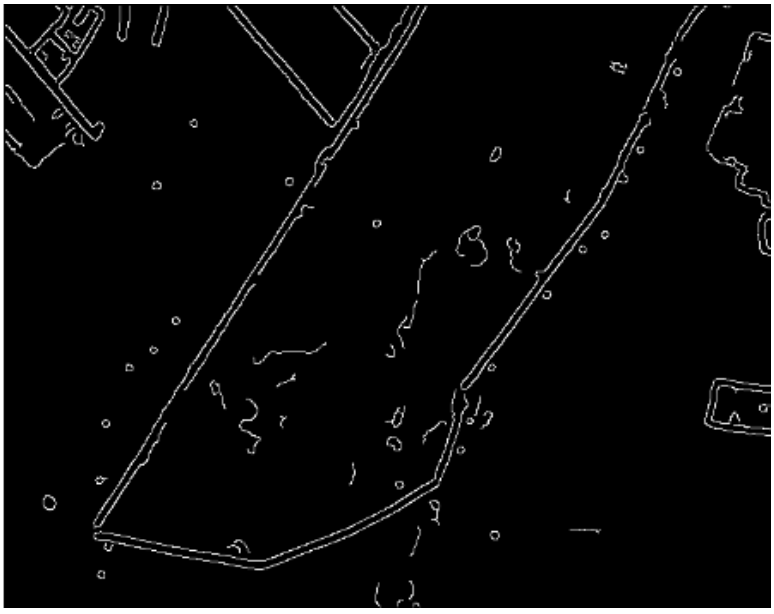
4. Pass a Gaussian filter over the gradient image to smooth the data (Figure 8).

Figure 8. Result of the Gaussian filter blurring the gradient image.



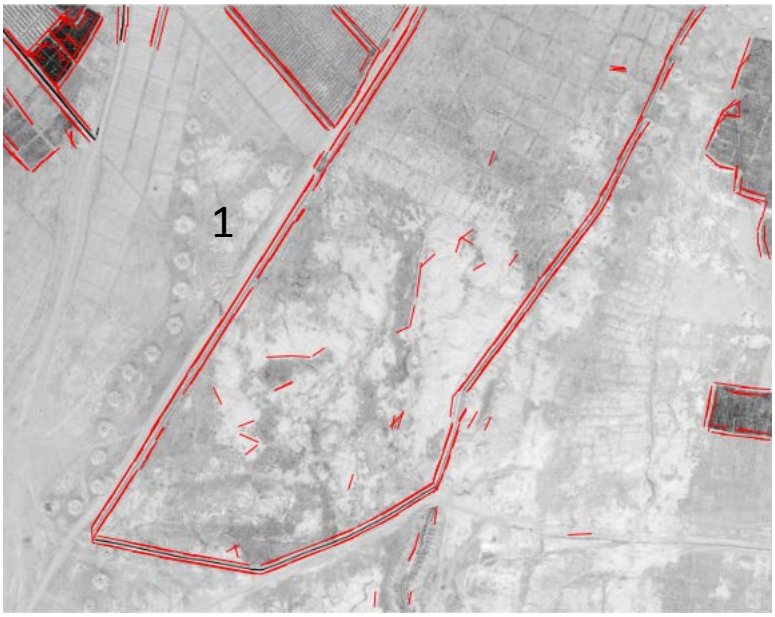
5. Pass the Canny edge-detection algorithm over the smoothed image (Figure 9).

Figure 9. Result of the Canny edge detector. The algorithm did a good job picking out the edges in the smoothed gradient image.



6. Use the Hough transform method to link the edges identified by the Canny edge detector (Figure 10).

Figure 10. Linear features identified by a Hough transform.




### 3.4.2 Methodology discussion

This methodology is able to detect the farm's fence line with fairly good accuracy compared to manually identifying the fence. However, this example highlights the Hough transform's poor performance in detecting corners connecting linear features, as seen by the gaps at the different bends in the fence. In addition, it highlights how the linear Hough transform is a poor detector of slightly curved linear features. Close inspection of the detected lines shows that a number of the long features are in fact several short features overlaid with each other. This is likely because the Hough transform is designed to look for *straight* linear features, which most real-life features are not. As mention previously, it is possible to define Hough transforms to look for any shape that can be parametrically described by an equation; so it may be possible to look for curved features given the proper equation. In addition, some formulations of the Hough transform include minimum distance criteria, which connects different segments within a specified distance of each other. Although this presents another parameter to optimize, this would reduce the number of line segments in the resulting image.

It is likely that the features extracted along the line labeled "1" are based on the edge of the dirt road and not on the actual fence. The road is lighter than the surrounding ground, which produces a distinct line in the gradient image (Figure 7) that the Hough transform identifies. In addition, the

other fence features produce lines in the gradient image because the fence has shadows in the original image (Figure 5). This illustrates a recurring drawback of these detection methods, which is that they work only if there is a distinguishable difference between the features and its surroundings. Otherwise, there is no way for the algorithm to differentiate between them.

Although there are some shortcomings to this methodology, the final result is useful and would help a user or analyst distinguish the layout and location of different features in this landscape. With information about the size of this landscape, additional functions could calculate distances between fence segments or estimate the area enclosed by the fence.

# 4 Repetitive-Feature Extraction

Linear and repetitive features are differentiated based on their appearance in the images of interest; however, real-life repetitive features often correspond to linear systems that are not apparent from a remote standpoint. For example, a series of individual telephone poles corresponds to wires that are often too thin to see in the imagery. Additionally, Karez well holes correspond to an underground system of tunnels that span many miles. An aircraft interacting with these features could be severely damaged such that it cannot complete its intended mission.

The following methodologies are examples of overall processes for finding repetitive features that can be applied to different types of features, regardless of what they are, and uses a number of methods explained in Section 2. Although the machine-learning procedure references Matlab functionalities, the general steps should be applicable to Python or other programming languages with image-processing capabilities.

## 4.1 Classification machine learning

The first methodology uses machine-learning algorithms. Machine learning is the process of training algorithms with known data to make predictions of new, unknown data (Schapire 2006). This is readily applicable to classification problems in which the algorithms classify data into a set of specified groups, such as classifying imagery data into different types of features. For repetitive-feature detection, this process consists of defining a model that adequately describes the target feature based on different properties, such as size, orientation, eccentricity, etc., and then training an algorithm to search for that model in new imagery data.

Matlab provides a variety of classifier algorithms, including decision tree, discriminant analysis, support vector machines (SVM), nearest neighbor, and ensemble classifiers, as defined in Table 7 (MathWorks 2015b).

The performance of each machine-learning algorithm depends on the characteristics of the image data being classified. One way to test algorithm performance is with a confusion matrix, which tests the algorithm's performance on the training dataset. For example, the algorithm is trained and then reimplemented on the training image to see if it extracts all the features it was supposed to find. The confusion matrix indicates in matrix

form the percentage of "true" or "false" features the algorithm identified correctly or incorrectly (MathWorks 2015a). The algorithm with the highest confusion matrix percentages was chosen for the repetitive-feature test case. Matlab provides a GUI program to make this kind of analysis more intuitive for the user; however, it would be possible to automate the process by calculating the confusion-matrix percentages for each tested algorithm and simply to pick whichever one had the highest success rate.

Table 7. Machine-learning classification algorithms (MathWorks 2015b)

| Classifier | Brief Description | Speed | Memory |
|---|---|---|---|
| Decision Tree | Trains the classification tree with test data and follows decisions in the tree to make new predictions | Fast | Low |
| Discriminant Analysis | Assumes the variables forming each class of data follow Gaussian distributions | Fast | Low to High |
| SVM | Finds a "hyperplane" that separates classes by the largest margin and uses the plane to classify new data | Medium to Slow | Medium to High |
| Nearest Neighbor | Calculates the distance between query points and nearest neighbors in training the dataset and classifies new data based on the distance between points | Medium to Slow | Medium |
| Ensemble | Combines other classifiers into one | Fast to Medium | Low to High |

### 4.1.1 Machine-learning methodology

The machine-learning process for detecting repetitive features begins with segmenting the image data and creating a training dataset of the features of interest. Many of the morphological operations and preprocessing techniques discussed in Section 2.1 are used to clean the imagery data and separate the features of interest from noise or other unwanted features. This requires a discernible characteristic between the feature and its surroundings, such as a difference in intensity values. Unfortunately, from the perspective of overhead remote imagery, telephone poles are typically too small to detect or are similar in color to the surrounding terrain. This methodology uses the poles' shadows for their discernible property as they have a different intensity value than the surrounding terrain. Detection of the actual poles would be more robust because shadows vary with time of day and cloud cover. Also, the shadows do not indicate the exact location of the telephone poles and corresponding wire; however, they are very close. Presumably any detected features will prompt a buffer zone that vehicles or personnel should avoid going into, and this zone would account for the slight offset between actual and detected pole locations.

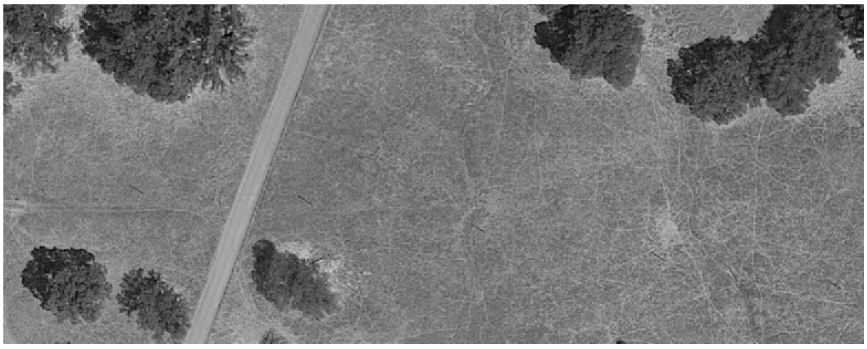
The following is an annotated step-by-step approach used to detect repetitive telephone pole shadows within an image via machine-learning methods.

1. Acquire the base image (Figure 11 in this case, from Google Earth).

Figure 11. Initial RGB image of a test site at FHL (JM-12).

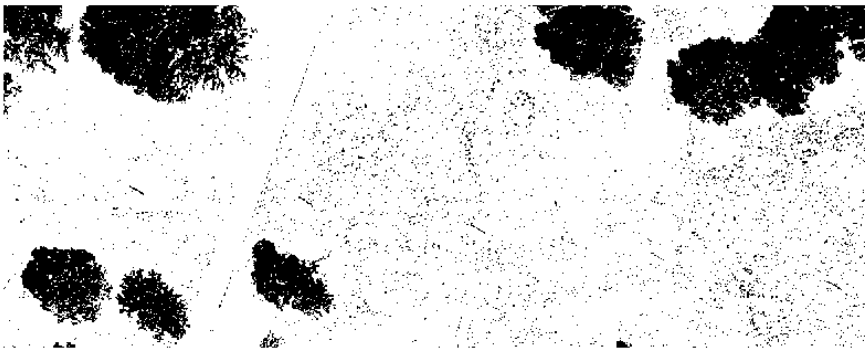
2. Convert the base image to grayscale (Figure 12).

Figure 12. Conversion of the RGB image to grayscale.



3. Convert the grayscale image to binary with Otsu's intensity threshold (Figure 13).

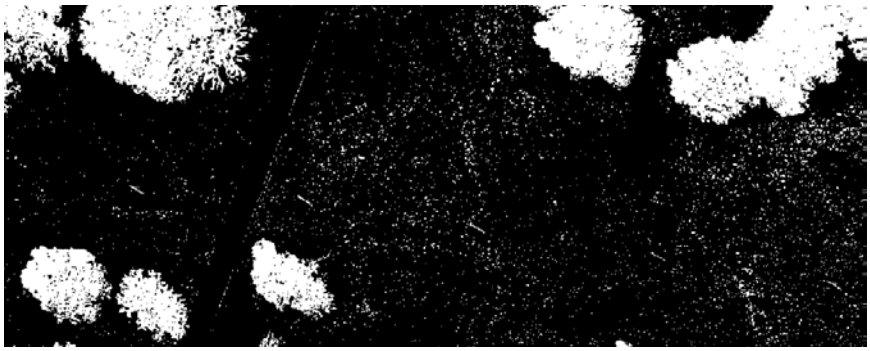
Figure 13. Conversion of the grayscale image to binary with the intensity threshold.



Now a series of morphological operators are used to clean up the binary image data. However, these act on the white portions of the image. Because the pole shadows are dark regions compared to their surroundings, they appear as black in the resultant binary image of Otsu's method (Figure 13). Taking the compliment of the binary image ensures that the features of interest are represented as white regions. Otherwise, the operators will act on the wrong portions of the image.

4. Take the compliment of the binary image (Figure 14).

Figure 14. Compliment of the initial binary image.



Next, a series of morphological operators are passed over the image to remove unwanted features while highlighting the desired ones. As mentioned in Section 2.1, a variety of morphological operations produces different results. The effectiveness of preprocessing depends on the combination and call order of these operations. It is important to use the same preprocessing procedure for both the algorithm training set and test dataset since the algorithm is trained to look for features with certain characteristics. If the test data is prepared differently than the training set, the algorithm will not be as effective.

Figure 15. Post morphological preprocessing.

Although it may not seem it, Figure 15 contains less noise and has cleaner features than the previous binary image (Figure 14). At this point, the image is segmented enough to produce a training dataset for the classifier algorithm. The training dataset is created by manually selecting the regions in Figure 15 that correspond to features of interest and calculating a variety of properties for those regions. This step uses the Matlab function called regionprops (Mathworks 2015c), which calculates a set of properties for each individual region within a binary image and then outputs them to a table. These properties include region area (in pixels), centroid location, orientation, perimeter (in pixels), and *many* more. These values construct a rigid model that describes the feature of interest in terms of specific properties that the algorithm can then search for. Repetitive features typically have similar characteristics as each other, which is useful for finding just those features in the image. For example, the shadows of each telephone pole in different images have similar orientations, eccentricity, and lengths as one another, as shown in Table 8.

Table 8. Model property values for identifying telephone-pole shadows.

| Test Site | Orientation (°) | Eccentricity | Length (pixels) |
|---|---|---|---|
| JM-12 | $-35° < \theta < -29°$ | E > 0.987 | 23 < L < 30 |
| JM-13 | $-29° < \theta < -21°$ | E > 0.990 | 24 < L < 39 |

Several more properties are calculated to provide as much data as possible to the classifier algorithm. The features highlighted in Figure 16 are those used as the training set for the classifier algorithm.

Figure 16. Extracted-feature training dataset of telephone poles at the JM-12 field-test site, FHL.

The actual training of the machine-learning algorithms is performed with the Matlab Classification Learner app. The input to this app is the property dataset for the training features, and the output is a function that takes image property datasets as input. Once the algorithm is trained, it can be tested on a brand new image. This requires preprocessing the new image and calculating the same set of properties as with the training image. This property dataset is then passed through the machine-learning function, which returns the identified features.

5. Plot identified features of interest on the input test image (Figure 17).

The last step is to display the detected poles. Points are plotted at the centroid of each shadow (as calculated with the regionprops function) to approximate the location of each telephone pole.

Figure 17. Detected telephone poles at the JM-13 field-test site, FHL.

### 4.1.2 Methodology discussion

The trained machine-learning algorithm effectively finds the telephone poles; however, it fails to find all of the pole shadows (Figure 18).

Figure 18. Actual telephone pole locations at test site JM-13, FHL.

The algorithm most likely fails to find two poles in particular because of the features adjacent to those specific poles. The left-most shadow intersects the dirt road, and inspection of the preprocessing result indicates that the program actually combines that shadow with the edge of the road. Thus, that particular feature does not fit the model for a telephone pole defined in the machine-learning algorithm, so it is not extracted. Similarly, the pole next to the tree combines with the tree during the preprocessing step, and is not identified.

This highlights the importance of using a good characteristic for separating the features from their surroundings. Unfortunately, it is extremely difficult to separate the actual telephone poles, at least in true-color format, from the surrounding terrain. The resolution is too coarse, and the surrounding terrain is nearly the same color brown as the telephone poles. As mentioned before, using shadows as the discernible feature is imperfect because their size, orientation, and visibility depend on the time of day and amount of sunlight when the image was taken. The success of this methodology should not depend on external variables such as weather or time of day. The determination of what characteristic to focus the algorithm on should be a case-by-case decision. The use of imagery metadata may assist in determining which characteristic or property to use in the machine-learning algorithm. For example, information about the date, time, and location of an image collection could guide an estimate of the sun's shadow length and orientation within that image.

Although there are some issues with this methodology, the result is useful and would help someone identify the location of telephone poles in this landscape. The issue is that this particular process is designed specifically for telephone-pole shadows with fairly specific orientations and sizes. This approach is not very robust and versatile in that sense.

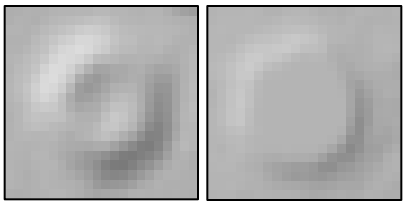
## 4.2 Template matching

In addition to machine-learning methods, template matching can identify repetitive features within a given image fairly quickly. This method finds strong matches between a template image and different regions of a larger image by calculating cross-correlations between the two datasets. Similar to the structure elements mentioned in Section 2.1, the template images are passed through the larger image pixel by pixel. For each template image location, a correlation is calculated between the pixels of the template and those of the target image. The result is a correlation map with different peaks and valleys indicating where there are strong or weak correlations between the two images. Peaks in the correlation field that are above a certain value are identified and extracted as matches. Multiple template images can be correlated with the same terrain image, which is useful if the feature of interest has multiple variations or if the user wants to identify different features in the same image.

### 4.2.1 Template-matching methodology

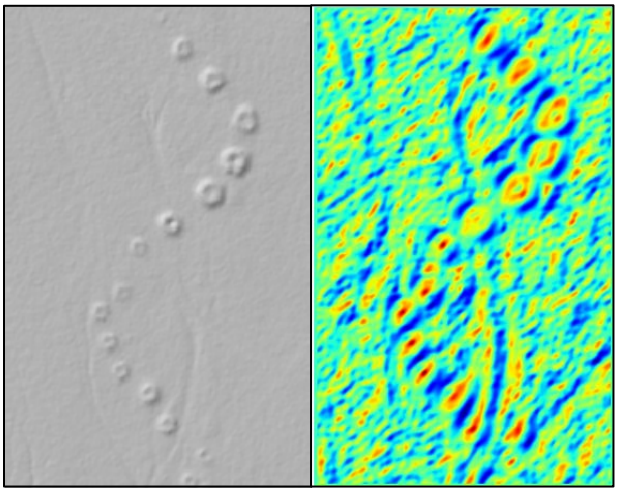
The basic template-matching methodology is fairly straightforward, but the following Karez hole example illustrates the different steps. First, template images of the feature of interest are selected (Figure 19). These templates are at the same resolution and size as the Karez holes in the terrain image (Figure 20).

Figure 19. Template images of Karez holes.



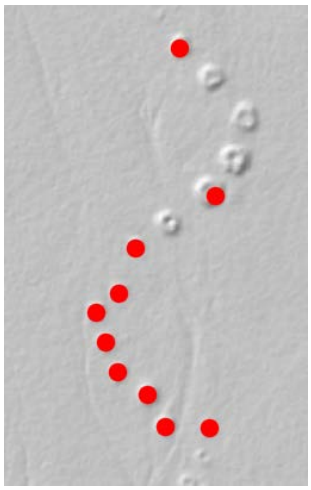
Next, each template image is cross-correlated with the larger terrain image (Figure 20, left), resulting in a cross-correlation field (Figure 20, right).

Figure 20. Example of a terrain image containing Karez holes (*left*) and the resultant cross-correlation field (*right*).



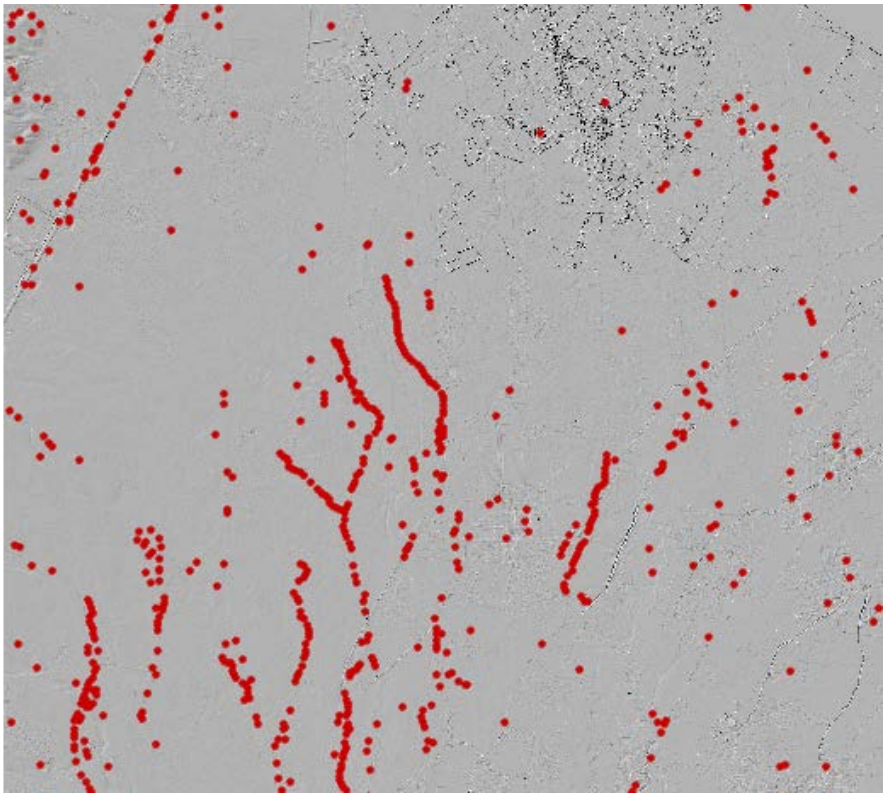
The next step is to extract the peaks in the cross-correlation field (Figure 21). This case considers the top 0.1% of peaks as matches.

Figure 21. Results of cross-correlating Karez hole templates in a landscape image.



This process operates quickly on large images, which is useful for identifying chains of Karez holes that may not have been obvious from such a large perspective (Figure 22).

Figure 22. Identification of Karez holes in a large-scale terrain image.



Although there are a number of false positives identified with this process, it gives a quick indication of where these chains are, how long they are, and which of them may be connected. It helps an analyst to hone in on a particular area where a number of those features have been detected while ignoring other areas where there are none or very few. The false positives are typically associated with the threshold value selected for the correlation tests. Different thresholds will lead to different amounts of true and false positives.

### 4.2.2 Scale-and-rotation-invariant template matching

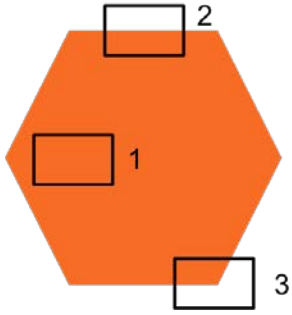
The cross-correlation method works well for symmetric features; however, it does not work well with features that are complex and appear at a variety of scales and orientations. Fortunately, there are multiple algorithms that are invariant to scale or rotation differences between the template image and a larger image. Unfortunately, a number of them are patent protected. SIFT (Scale-Invariant Feature Transform) has been widely used for years, and SURF (Speeded-Up Robust Features) is a faster variation of SIFT; but both are patented. There are a few open-source methods available, including Ciratefi (Kim and Alves de Araujo 2007) and ORB (Oriented

FAST and Rotated BRIEF) (Rublee et al. 2011). This paper will briefly discuss some of the features of ORB as it is provided in OpenCV.

#### *4.2.2.1 Oriented FAST and Rotated BRIEF (ORB)*

As the full name suggests, ORB is based on two different algorithms: FAST (Features from Accelerated Segment Test) and BRIEF (Binary Robust Independent Elementary Features). FAST identifies keypoints, and then BRIEF describes these points in a way that the computer can understand and then search for. The following explanation of keypoints is adapted from (Mordvintsev and K 2013). Keypoints are regions of an image that are unique and do not appear in more than one spot in that image. These could be corners of features with interesting interfaces between multiple colors or intensity values. Consider Figure 23.

Figure 23. Illustration of feature keypoints.




Region 1 in Figure 23 cannot be a keypoint because the entire region is the same color. The contents of that box would look the same anywhere inside the hexagon. Region 2 cannot be a keypoint because it is simply a flat interface between two colors. Moving that box along the upper edge of the hexagon would have no effect on the contents of the box. However, region 3 could be a keypoint because it contains a unique corner. Although there are other corners of the hexagon, a similar box over those corners would look different from the region 3 box above. An algorithm that identifies keypoints looks for regions in the image where slight box movements cause large differences between the premovement and postmovement regions (Mordvintsev and K 2013).

Once the keypoints are identified with FAST, they must be described numerically in a way that the computer can compare against the descriptions of other regions. This is where the BRIEF portion of ORB comes into ef-

fect. However, BRIEF does not handle rotations well, so the keypoints' orientations are first calculated based on the intensity distribution of pixels in a small circular area around each keypoint (Mordvintsev and K 2013). This step assumes that the keypoint's location differs from the centroid of the intensity-weighted moment of that small circular region. This is analogous to how the centroid of a three-dimensional object's mass moment of inertia may differ from that object's geometric centroid due the object's shape and mass distribution. If these two points differ in location, then a vector can be drawn from one to the other, thus giving an orientation to each keypoint (Rublee et al. 2011). ORB uses these keypoint orientations to steer the BRIEF portion of the algorithm, which improves its performance (Mordvintsev and K 2013).

SIFT and SURF create 128 and 64 dimension floating-point vectors, respectively, to describe each keypoint, which uses a large amount of computational memory. Statistical procedures, such as principal component analysis, can reduce the number of dimensions needed to fully describe each keypoint. In addition, converting this data to binary string format via hashing functions reduces its memory usage further. BRIEF improves this process by producing the binary strings for each feature directly without constructing the descriptive vectors first. These binary strings are then matched by calculating the Hamming distance between each pair of strings (Calonder et al. 2010). The Hamming distance between two binary strings is the number of coefficients that differ between them (Symonds 2007). Small Hamming distances indicate two strings that have similar coefficients, which indicates they match closely. This implies that the two keypoints corresponding to those strings have similar descriptions and thus are similar to each other.

The accuracy of matched points is assessed based on a nearest-neighbor test taken from Lowe (2004). For each identified keypoint, this test compares the distances to both its closest and second closest matches. If the ratio of the closest distance to the second closest distance is less than 0.8, then it is considered a correct match. As explained by Lowe (2004), "this measure performs well because correct matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve reliable matching." (Lowe (2004) finds that this threshold removes 90% of false matches while removing less than 5% of correct matches. Based on the number of correct matches found, the user can determine whether the feature of interest has actually been identified. For example, if there are

ten correctly matched keypoints between a prescribed target feature and a new image, the user might assume that their feature was found in that new image.

#### *4.2.2.2 ORB template-matching example*

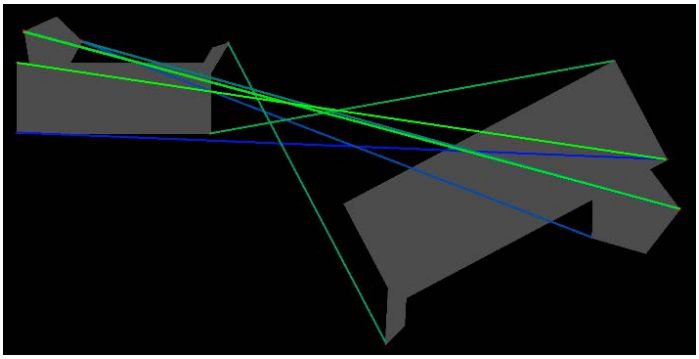
The following example illustrates some of the steps associated with using ORB to detect features with scale and rotation variations. The first step is to identify keypoints for the feature of interest (Figure 24).

Figure 24. Keypoints identified on a geometric shape.

ORB identifies a number of keypoints around the feature of interest, including several that are bunched together at the same location. As expected, the chosen keypoints are at the corner of the shape as these locations provide unique features of the shape.

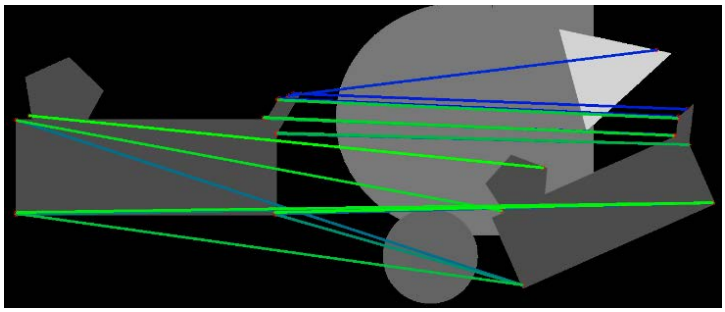
Next, ORB is tested with a rotated and scaled version of the base feature. The matches between the base shape and a rotated and scaled version are linked with colored lines in Figure 25.

Figure 25. Linked keypoints between scaled and rotated versions of the same feature using ORB.



ORB does a good job of linking the correct keypoints between the base image and the altered version. The only points that are incorrectly matched are those for the 90° corners, which are difficult to track because there are multiple corners like that in the feature. To truly test ORB, an altered version of the base image was mixed in with other shapes, as in Figure 26.

Figure 26. Linked features between the base image and mixed image.



ORB handles this test fairly well, but the addition of other shapes leads to some incorrect matches. Additional tests indicate that ORB tracks the feature better if the rotation between the base image and the mixed image is small. ORB performs better in these situations if the additional shapes are rounded while the feature of interest has sharp corners or vice versa.

Figure 27 and Figure 28 illustrate matched keypoints identified with ORB for a few different kinds of shapes.

Figure 27. Ten matched keypoints for an amorphous shape.

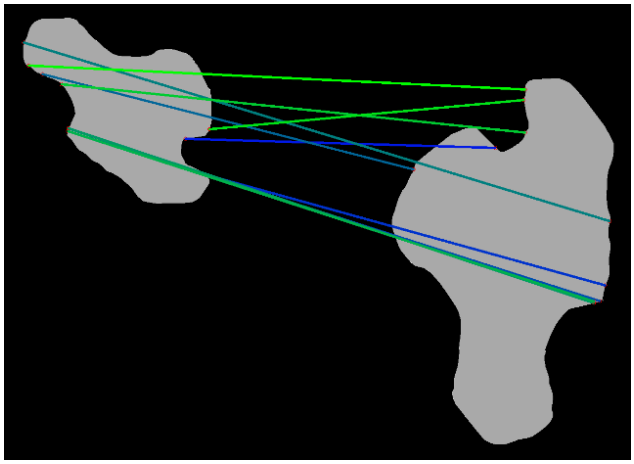
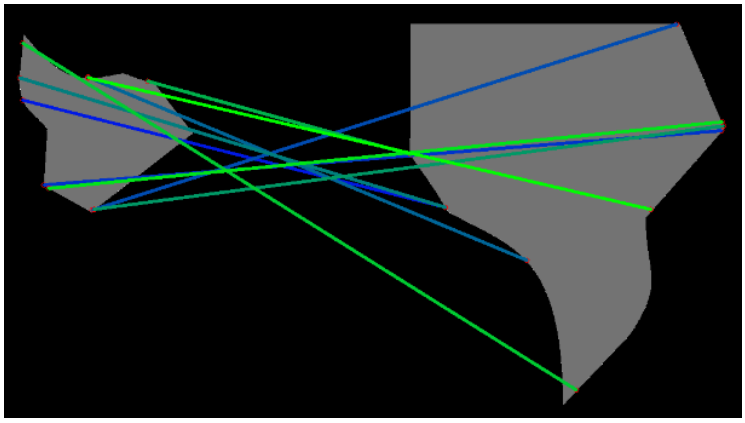


Figure 28. Ten matched keypoints for a shape with geometric and amorphous components.



Each image has more matched keypoints than those plotted, but only a small subset were plotted to reduce clutter. Comparing these figures with Figure 25 indicates how ORB produces better results for geometric shapes than amorphous shapes. This makes sense as the geometric shape has several distinct corners, which generally make strong keypoints. This proves to be a limiting factor of ORB or any feature detection algorithms that follows a similar methodology. It would be difficult to accurately track features with curved features.

### 4.2.3 Methodology discussion

Template matching is an intriguing method for identifying features because of how quick it is to implement. Figure 22 illustrates how the cross-correlation method can quickly identify a number of features within a large image. Although a number of the identified features are incorrect, the result still provides a useful starting point for analysts to target their efforts. As with most of these methods, users must have some prior knowledge of the feature they hope to detect. If they can extract a number of templates for different features of interest from a larger image, then it is very simple to use this cross-correlation method to identify a number of those features across the entire large image. However, the method is still dependent on an arbitrary selection of a correlation threshold value and is limited to scale-and-rotation-invariant situations.

ORB proved adequate at detecting scaled and rotated versions of objects, but it has limitations when it comes to curved or rounded features. It also has limitations when identifying multiple features within the same image.

Searching for several features at different scales and orientations is computationally intensive, so this method is likely reserved for situations that look for only one or two features.

Both methods achieve their intended goals of identifying and matching features; however, they do not operate with the level of precision that the Army likely requires. As with the detection methods previously discussed, these tools provide good guidance tools but not autonomous detection methods.

## 4.3 Comparison of repetitive-feature methodologies

Machine learning and template matching have their own advantages and disadvantages when it comes to extracting repetitive features. Machine-learning algorithms are intensive to train but are simple to use after proper setup. Template matching is quick and simple to use but can identify a number of false positives and is not very robust. Both methods are able to achieve feature extraction; however, their effectiveness depends on creating comprehensive yet specific models for each method to search for, which is difficult to perfect.

A recurring theme in this paper is the importance of proper preprocessing and segmentation. This step is tantamount to successful feature detection with machine learning. As mentioned previously, any new image on which the machine-learning algorithm is implemented must be preprocessed the exact same way as the image used to train the algorithm. The results of preprocessing are dependent on characteristics of the image itself, so this approach may fail to find features in disparate images that have varying preprocessing results. In addition, the algorithms are trained to look for features with rather specific characteristics, which could prove problematic if there are variations in the feature of interest.

Template matching does not require preprocessing as machine learning does; however, it requires template images that adequately cover all variations of the feature of interest. If a Karez hole in the terrain image has a filled-in hole and the template Karez image has a hole in it, then the cross-correlation method may fail. In addition, ORB and other scale-and-rotation-invariant template-matching schemes are able to account for variations in the feature of interest, but it is unrealistic to implement such methods on a large image due to the computational expense.

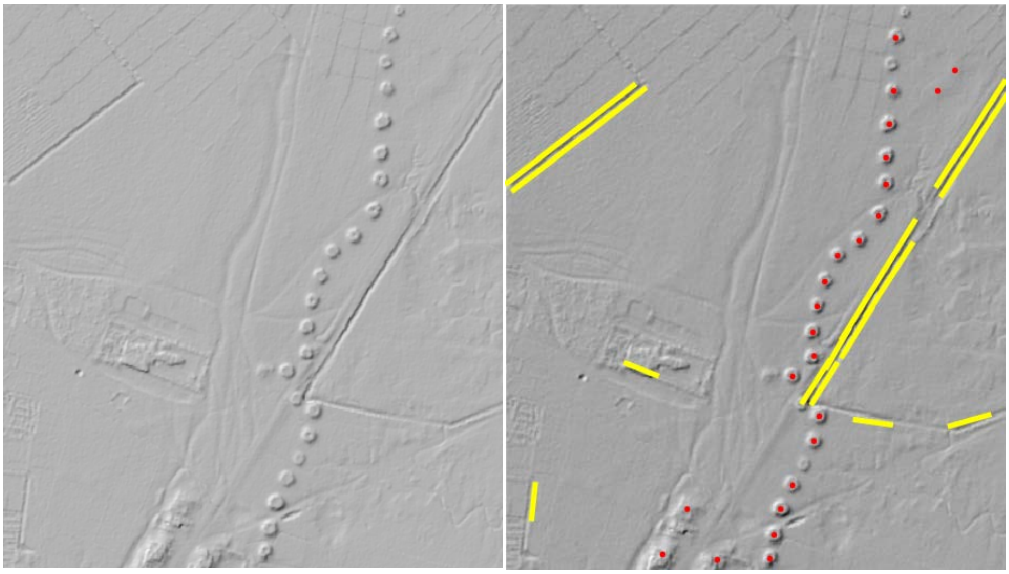
A major drawback to both repetitive-feature methodologies is that they require prior knowledge about the features of interest. These techniques cannot be used to find unknown features that happen to repeat in the data. That capability would be extremely useful for Army initiatives; however, it is a very difficult problem and requires more-advanced methods and mathematical techniques than those explored here (Ferreira and Pinho 2014).

In summary, both methods can find features, but they require very specific conditions to do so. In addition, they require a decent amount of user interaction to achieve useful results. These methods are unable to achieve completely autonomous feature detection as is.

# 5 Combination of Linear- and Repetitive-Feature Detection Methods

The methods for detecting linear and repetitive features are not exclusive to each, and it is possible to implement them on the same image. For example, the left image in Figure 29 contains Karez holes, a fence line surrounding a farm, and some structures.

Figure 29. Landscape containing both linear and repetitive features.



Implementing the Hough transform method to detect fences and then the cross-correlation method to find Karez holes provides an analyst a useful overview of what is in this landscape and their geospatial distributions. Although Figure 29 does not highlight all of the features in the image, it illustrates some of the capabilities of combining these methods on the same image data.

# 6 Future Directions

The field of image processing is vast with far more methods and algorithms than those described in this paper. This section provides a brief overview of a few additional areas that may be of interest for studies with similar goals, as well as research areas that can improve autonomous feature detection.

## 6.1 Combination of machine learning and template matching

Machine learning and template matching were used separately to identify repetitive features in this study. However, they may prove useful if used in conjunction with each other. Selecting the features detected by both methods would increase the fidelity of the final result and might be a good way to remove false positives detected by either scheme.

In addition, training machine-learning algorithms requires a training dataset of manually selected features, which is time consuming and requires user interaction. Quickly running an image through a template-matching scheme might provide a useful set of initial feature locations, which may speed the construction of the training set.

## 6.2 Elevation-data fusion

Current remote sensors can measure elevation data, which could be very useful in extracting features from a landscape. In the same way that features have certain visual properties, they may have distinct elevation characteristics. It may be easier to identify Karez holes based on their elevation footprint than their visual appearance. If the elevation data is fine enough resolution, it may contain information related to telephone poles, which would alleviate the issue of using the telephone pole's shadows to detect them. In this regard, elevation data would add a whole new dimension to these types of analyses.

## 6.3 Context-based processing

Accounting for the geospatial distribution of the features detected with each scheme may improve the final set of detected features. For example, if an algorithm with a high "true" classification rate (true and false positives) returns a large set of possible features, additional functions based on the distance between points might be able to remove the false positives.

For example, features that are far from any other could be ignored as noise. Other geospatial functions might test the linearity of features, the shape they create, or other patterns.

## 6.4 Color spaces and topology

Color images are depicted based on different color spaces, which use numbers and different components to define all possible colors in an image. The most common color space is RGB, which uses a three-number set indicating the levels of red, green, and blue of each pixel. It is possible to segment an image based on each color component, which occasionally highlights new information, depending on the feature of interest's color. There are other color spaces including HSV (hue, saturation, value), CMYK (cyan, magenta, yellow, key [black]), or Lab (lightness, a [green–red], b [blue–yellow]) (Farley 2010). In the same way that an RGB image can be segmented based on one of its three components, segmenting an image in one of these additional color spaces may highlight different aspects of the image or feature. Additional information on color segmentation is available in Gonzalez and Woods (2002).

Point set topology is a field of mathematics that can describe the spatial proximity of objects via properties that remain unaffected by image stretching, deformation, or twisting (Niccolai et al. 2010). These methods may be useful for autonomously relating features that appear at different scales and orientations within different images. Additionally, topological methods may improve the Hough transform results by connecting individual line features that are close to each other. These are worth additional investigation.

## 6.5 Multispectral imagery

Multispectral and hyperspectral imagery data contain much more information than panchromatic image data does. In some cases, a particular feature of interest may be more obvious in one band than in another. As mentioned previously, many of the methods in this paper require input data in a grayscale format. This means that each band of multispectral or hyperspectral data would need to be processed individually using these methods. The additional information gleaned from several different bands could offset the additional computational expenses of processing several different bands.

## 6.6 Image metadata

Similar to multiple spectral bands of data, metadata provides additional information regarding the data in an image. As mentioned previously, metadata could guide the training of a particular machine-learning algorithm by indicating which feature characteristics are best. For example, if a particular image's metadata includes the date, time, latitude, longitude, and weather during data collection, then it should be possible to estimate sun exposure and to calculate the direction of any shadows at that location. Identifying a shadow could be very useful even if the geometry of the object is unknown; just the knowledge of its presence as a vertical object is useful. In addition, the length of the detected shadows could potentially indicate the height of those features, which is also useful information.

## 6.7 Stereoscopics

Stereoscopic images are based on stereopsis, which is the perception of depth due to multiple images of the same object that are slightly offset from one another. Slight variations between the two images are indicative of the elevation of components in those images. Humans' depth perception is attributable to this phenomenon (Hubel 1995). If elevation data is unavailable for a particular application, it may be possible to extract elevations from multiple images via stereopsis. The challenge with this is being able to extract differences between the two images and then quantifying them in some meaningful manner.

## 6.8 Additional applications

This report focuses on using feature detection methods for preventative measures or when planning operations through a certain area. However, these same techniques could help to detect features for additional investigation. For example, these methods might help military analysts identify communications equipment and installations, "cookie-cutter" residential compounds, or tracks through a landscape, etc.

## 6.9 Preprocessing and segmentation methods

The overall success of many feature detection methods depends on how well the image is preprocessed or segmented, which signifies the importance of this step in the feature detection process. However, for many current preprocessing methods, there are simply too many choices for the

user to make and understand. For example, combining the methods mentioned in the "Data Preprocessing" section (Section 2) will have different results depending on the order in which they are applied to an image. In addition, several of these functions have inputs that tweak their results, which require optimization or a user's specification. Many of these inputs depend on the analyzed image, and some that work great for one type of image analysis may not work well for others. It may be possible to develop a preprocessing scheme for a specific type of image and analysis, but that would require a fairly large amount of time to thoroughly develop.

Development of robust preprocessing and segmentation methods is a common problem in the field of image processing, and stating the need for them may be redundant and obvious, but this study serves as yet another case for improved techniques. The high-fidelity requirements of military operations require image-processing results that are much more precise than those that are currently achievable. Improved preprocessing will significantly help autonomous feature detection by providing the detection methods with clean and properly formatted image data.

# 7 Conclusions and Recommendations

This report serves as a synopsis of basic methods used to segment and extract different types of features from imagery. Several powerful and advanced tools can effectively detect different types of features. However, the variety of methods highlights one of the biggest hurdles when it comes to feature detection—there is no one-size-fits-all way to confidently detect all the features within an image that a military user may be interested in. It is likely that each situation will require its own customized version of these methods, especially those requiring preprocessing. With the current state of preprocessing and segmentation, these techniques and tools may speed up an analyst's search for features, but it is unlikely they could fully replace a human's visual recognition skills. Nearly every step requires some user interaction to produce usable results. Advancements in this field will lead to more intelligent algorithms at which point the military might be able to rely on image processing for this task. But at the present state, the user interaction required to effectively run these methods is probably as involved and intensive as an image analyst manually going through each image.

Although these methods cannot fully replace a human's ability to detect features in an image, as is, they are still useful aids. The Karez hole example in the "Template-matching methodology" section (Section 4.2.1) illustrates how these methods can quickly identifying large numbers of features. Also, a number of the filters in the "Data Preprocessing" section (Section 2), such as gradient or Laplacian filters, can highlight features in the data that are not initially obvious to the human eye. These tools can supplement the feature detection process and help military image analysts identify potential threats, unobstructed tracts of land, or other features of interest.

# References

Argialas, D. P., and O. D. Mavrantza. 2004. Comparison of Edge Detection and Hough Transform Techniques for the Extraction of Geologic Features. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 34: 790–795.

Beucher, S. 2010. *Image Segmentation and Mathematical Morphology*. MINES ParisTech. http://cmm.ensmp.fr/~beucher/wtshed.html (accessed 11 January 2016).

Burger, W., and M. J. Burge. 2009. *Principles of Digital Image Processing—Fundamental Techniques*. London: Springer.

———. 2013. *Principles of Digital Image Processing*. London: Springer.

Calonder, M., V. Lepetit, C. Strecha, and P. Fua. 2010. BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the 11th European Conference on Computer Vision (ECCV), Heraklion, Crete, Creece, 5–11 September*, ed. K. Daniilidis, P. Maragos, and N. Paragios, 778–792.

Claypoole, R., J. Lewis, S. B., and K. Kelly. 1997. Laplacian Edge Detection. ELEC 539: Image Morphing. Houston, TX: Rice University, Digital Signal Processing Group. http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/laplacian.html (accessed 24 March 2015).

Delmas, P. 2015. Prewitt Filter. Auckland, New Zeland: University of Auckland, Department of Computer Science. https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Prewitt_2up.pdf (accessed 25 March 2015).

Eddins, S. 2002. The Watershed Transform: Strategies for Image Segmentation. *MathWorks*. http://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html?refresh=true (accessed 11 January 2016).

Efford, N. 2000. Morphological Image Processing. *Digital Image Processing: A Practical Introduction Using Java*. Pearson Education. Lecture notes online. Auckland, New Zeland: University of Auckland, Department of Computer Science. https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm#refs.

Farley, J. 2010. A Short Guide to Color Models. *Sitepoint*, 27 June. https://www.sitepoint.com/a-short-guide-to-color-models/.

Ferreira, P. J. S. G., and A. J. Pinho. 2014. A Method to Detect Repeated Unknown Patterns in an Image. In *11th International Conference on Image Analysis and Recognition*, Vilamoura, Portugal, 22–24 October, ed. M. Kamel. Springer.

Fisher, R., S. Perkins, A. Walker, and E. Wolfart. 2003. Digital Filters. Edinburgh, Scotland: University of Edinburgh, School of Informatics. http://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm (accessed 18 March 2015).

Fitton, N. C., and S. J. D. Cox. 1998. Optimising the Application of the Hough Transform for Automatic Feature Extraction from Geoscientific Images. *Computers and Geosciences* 24 (10): 933–951.

Gonzalez, R. C, and R. E. Woods. 2002. *Digital Image Processing*. 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Grady, L. 2006. Random Walks for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (11): 1768–1783.

Hamarneh, G., K. Althoff, and R. Abu-Gharbieh. 1999. Automatic Line Detection. Chalmers University of Technology, Department of Signals and Systems. https://www.cs.sfu.ca/~hamarneh/ecopy/compvis1999_hough.pdf (accessed 30 March 2015).

Hubel, David. 1995. Stereopsis. *Eye, Brain, and Vision*. W. H. Freeman. Online at Harvard Medical School. http://hubel.med.harvard.edu/book/b36.htm.

Kim, H. Y., and S. Alves de Araujo. 2007. Grayscale Template-Matching Invariant to Rotation, Scale, Translation, Brightness and Contrast. In *Advances in Image and Video Technology*, ed. D. Mery and L. Rueda, 4872:100–113. Berlin: Springer-Verlag Berlin Heidelberg.

Lowe, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60 (2): 91–110.

Maini, R., and H. Aggarwal. 2010. A Comprehensive Review of Image Enhancement Techniques. *Journal of Computing* 2 (3): 8–13.

MathWorks. 2015a. Assess Classifier Performance. *MathWorks*. http://www.mathworks.com/help/stats/assess-classifier-performance.html.

———. 2015b. Choose a Classifier. *MathWorks*. http://www.mathworks.com/help/stats/choose-a-classifier.html.

———. 2015c. Regionprops. *MathWorks*. http://www.mathworks.com/help/images/ref/regionprops.html.

Mavrantza, O. D., and D. P. Argialas. 2003. Implementation and Evaluation of Spatial Filtering and Edge Detection Techniques for Lineament Mapping—Case Study: Alevrada, Central Greece. In *Proceedings of SPIE 4886, Remote Sensing for Environmental Monitoring, GIS Applications, and Geology II*, ed. M. Ehlers.

Mirmehdi, M., G. A. W. West, and G. R. Dowling. 1991. Label Inspection Using the Hough Transform on Transputer Networks. *Microprocessors and Microsystems* 15: 167–173.

Mordvintsev, A., and A. R. K. 2013. *OpenCV-Python Tutorials*. http://opencv-python-tutroals.readthedocs.org/en/latest/index.html (accessed 18 December 2015).

Morse, Bryan S. 2000. Lecture 4: Thresholding. Brigham Young University. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf (accessed 16 September 2015).

Musoromy, Z., S. Ramalingam, and N. Bekooy. 2010. Edge Detection Comparison for License Plate Detection. In *Proceedings of the 11th International Conference on Control, Automation, Robotics, and Vision.*

Niccolai, A., M. Niccolai, and C. D. Oliver. 2010. Point Set Topology Extraction for Branch and Crown-level Species Classification. *Photogrammetric Engineering and Remote Sensing* 76 (3): 319–330.

Quackenbush, L. J. 2004. A Review of Techniques for Extracting Linear Features from Imagery. *Photogrammetric Engineering and Remote Sensing* 70 (12): 1383–1392.

Rahnama, M, and R. Gloaguen. 2014. TecLines: A Matlab-Based Toolbox for Tectonic Lineament Analysis from Satellite Images and DEMs, Part 1: Line Segment Detection and Extraction. *Remote Sensing* 5938–5958.

Rothwell, C. A., J. L. Mundy, W. Hoffman, and V.-D. Nguyen. 1995. Driving Vision by Topology. In *Proceedings of IEEE International Symposium on Computer Vision (ISCV '95), Coral Gables, FL*, 395–400.

Rublee, E., V. Rabaud, K. Konolige, and G. Bradski. 2011. ORB: An Efficient Alternative to SIFT or SURF. *ICCV '11 Proceedings of the 2011 International Conference on Computer Vision,*. 2564–2571. Washington DC: IEEE Computer Society.

Ryerson, C. C., and J. McDowell. 2008. Anywhere - Anytime: Enhancing Battlespace Vertical Mobility. Proposal to the American Institute of Aeronautics and Astronautics. Hanover, NH: U.S. Army Engineer Research and Development Center.

Schapire, R.. 2006. Machine Learning Algorithms for Classification. Princeton University, Department of Computer Science. http://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf.

Scikit-Image. 2011. *Straight line Hough Transform*. Scikit-Image Development Team. http://scikit-image.org/docs/dev/auto_examples/edges/plot_line_hough_transform.html.

Symonds, P. 2007. MATH32031: Coding Theory. Part 2: Hamming Distance. Manchester, UK: University of Manchester, School of Mathematics. http://www.maths.manchester.ac.uk/~pas/code/notes/part2.pdf.

Vilnius University. 2009. Derivative-Based Operations. http://www.mif.vu.lt/atpazinimas/dip/FIP/fip-Derivati.html (accessed 24 March 2015).

Wang, S., F. Ge, and T. Liu. 2006. Evaluating Edge Detection through Boundary Detection. *EURASIP Journal on Applied Signal Processing*, 1–15. Hindawi Publishing Corporation.

Wittman, T. 2010. Image Processing in the Fourier Domain. Los Angeles: University of California, Los Angeles (UCLA), Department of Mathematics. http://www.math.ucla.edu/~wittman/PCMI/fourier_pcmi6.pdf (accessed 18 March 2015).

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| **1. REPORT DATE** *(DD-MM-YYYY)* | **2. REPORT TYPE** | **3. DATES COVERED** *(From - To)* |
|---|---|---|
| April 2017 | Technical Report/Final | |

**4. TITLE AND SUBTITLE**

Linear- and Repetitive-Feature Detection Within Remotely Sensed Imagery

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Brendan A. West

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**
9K3D08

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

U.S. Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

**8. PERFORMING ORGANIZATION REPORT NUMBER**

ERDC/CRREL TR-17-6

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Army Terrestrial Environmental Modeling and Intelligence System (ARTEMIS)
U.S. Army Engineer Research and Development Center (ERDC)
Cold Regions Research and Engineering Laboratory (CRREL)
72 Lyme Road
Hanover, NH 03755-1290

**10. SPONSOR/MONITOR'S ACRONYM(S)**
ARTEMIS

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

The United States Army has a variety of applications for identifying features of interest within remote imagery. Whether it is characterizing a landscape while planning operations or trying to find particular installations in an urban setting, the Army can glean a significant amount of information from imagery data. This study investigates methods that can detect linear and repetitive features contained in remotely sensed images that are in panchromatic or true-color formats. Image-processing techniques, including Hough transforms, machine learning, and template matching, are capable of detecting different kinds of features within images. However, the success of these methods depends on effectively preprocessing image data, which has proven difficult and intensive for certain images. In many cases, the amount of user interaction needed to produce useful results exceeds the amount of labor needed to manually inspect individual images. At their current state, these methods provide useful tools to help analysts detect features but do not replace their expertise. This report summarizes several techniques for preprocessing image data and then detecting linear and repetitive features in that data.

**15. SUBJECT TERMS**
Geographic information system, Geospatial Remote Assessment for Ingress Locations, GRAIL, Linear features, Remote image processing, Remote feature detection, Remote-sensing images, Repetitive features

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** |
|---|---|---|---|---|---|
| **a. REPORT** | **b. ABSTRACT** | **c. THIS PAGE** | | | **19b. TELEPHONE NUMBER** *(include area code)* |
| Unclassified | Unclassified | Unclassified | SAR | 62 | |